A Synthetic Penalized Logitboost to model Mortgage Lending with Imbalanced Data

Jessica Pesantez-Narvaez, Montserrat Guillen and Manuela Alcañiz*

Department of Econometrics, Riskcenter-IREA, Universitat de Barcelona, 08034 Barcelona, Spain; jessica.pesantez@ub.edu, 0000-0003-3161-7807 (J.P.-N.); mguillen@ub.edu, 0000-0002-2644-6268 (M.G.)
*Corresponding author: malcaniz@ub.edu, 0000-0002-5028-1926 (M.A.) Tel.: +34-934-021-983[1]

Abstract

Most classical econometric methods and tree boosting based algorithms tend to increase the prediction error with binary imbalanced data. We propose a Synthetic Penalized Logitboost based on weighting corrections. The procedure (i) improves the prediction performance under the phenomenon in question, (ii) allows interpretability since coefficients can get stabilized in the recursive procedure, and (iii) reduces the risk of overfitting. We consider a mortgage lending case study using publicly available data to illustrate our method. Results show that errors are smaller in many extreme prediction scores, outperforming a number of existing methods. Our interpretations are consistent with results obtained using a classic econometric model.

Keywords: Imbalanced, boosting, interpretation, prediction, binary.

JEL Classification

C01, C02, C13, C60.

1. Introduction

Predicting binary decision problems is important in empirical economics. For instance, identifying whether an applicant will default in future or be turned down under the Home Mortgage Disclosure Act[2] (HMDA) contributes to the study of financial inclusion policy. In fact, the notion of having events versus non-events (a binary response) can be the result of a latent and unobserved random variable that triggers an event when it is high enough, so that extreme values then turn into event responses.

Class-imbalanced data are relevant primarily in the context of supervised machine learning involving two (dichotomous) or more classes. *Imbalanced* means that the number of observations is not the same for each class of a categorical variable, in other words, one class is represented by a large number of observations while the other is represented by only a few (Japkowicz and Stephen, 2002).

In the context of mortgage lending, for example, Munnell et al. (1996) have dealt with an imbalanced class problem. They found that black and Hispanic applicants were more likely than whites to be denied mortgage loans. Thus, the class corresponding to the applicants who were denied was much smaller than the applicants who were approved. The minority class (denied

[1] The authors confirm that this paper has not been submitted elsewhere.
[2] HMDA is a disclosure law that provides publicly available information on the US mortgage market where applicants' characteristics are registered in order to identify possible patterns of discriminatory lending. The 94th United States Congress found that some financial institutions tend to decline qualified applicants without sufficient rationale.

mortgage lending) could be coded as one, while the majority class (approved for mortgage lending) could be coded as zero.

There is evidence that the prediction accuracy of this type of events seems to remain problematic. King and Zeng (2001) note that classical econometric methods can underestimate the probability of occurrence in the minority class, while Krawczyk (2016) finds that machine-learning methods tend to exhibit a bias towards the majority class.

There is a vast literature devoted to proposing techniques to handle the class imbalance problem. Barandela et al. (2003), Kotsiantis et al. (2006), Longadge et al. (2013), and Lin et al. (2017) summarize four types of techniques:

(i) data preprocessing (balancing the data by oversampling, which increases the number of observations in the minority class, or by undersampling, which reduces observations in the majority class) using an algorithm approach (creating or modifying algorithms with the threshold and one-class learning methods),
(ii) cost-sensitive solutions (minimizing the costs of misclassification),
(iii) feature selection (finding the optimal combination of covariates that gives the best classification), and
(iv) resampling techniques incorporated in classifier ensembles such as boosting or bagging, which have given risen to proposals such as Synthetic Minority Oversampling (SMOTE) (Chawla et al., 2002), RUSBoost (Seiffert et al., 2009), UnderBagging (UB) (Barandela et al., 2003), and OverBagging (Wang and Yao, 2009).

In our view, however, the modelling and interpretability of imbalanced class phenomena in a joint process without overfitting data remains a subject beyond the scope of machine learning. We propose a Synthetic Penalized Logitboost that aims to decrease the mean square error in the highest and lowest prediction scores of the probability of minority class occurrence, by introducing a weighting mechanism that recalibrates a Logitboost to reduce the risk of overfitting. The Synthetic Penalized Logitboost improves the detection of extremes in the data if the purpose is to look for unusual patterns rather than for average cases. For this purpose, we borrow the specification of the model put forward by Munnell et al. (1996) to predict mortgage loan denial with a logistic regression.

The paper is divided into five sections after the introduction. Section 2 describes the theoretical framework that motivates the paper. Section 3 describes the methodology in detail, specifically logistic regression (econometric model for binary prediction), Logitboost, Gradient Tree Boost (boosting-based machine learning for binary prediction) and the proposed algorithm. Section 4 describes the data set used in an illustrative example. Section 5 sets out the results and predictive performance measured by the root-mean-square error and includes the model's interpretation. Finally, Section 6 contains the conclusions.

2. A Closer Look at the Theoretical Framework

Considering a supervised statistical learning framework, let us start from a data set of n observations with a quantitative target variable (dependent variable) Y_i, $i=1,..,n$ that has some relationship with a set of P predictor variables denoted as X_{ip}, $p=1,...P$ (also known as covariates). This can be written as:

$$Y_i = F\left(X_{ip}\right) + \epsilon_i , \tag{1}$$

where F is a deterministic function of the X_{ip}, and ϵ_i is the error or disturbance term that captures the influence of omitted factors, is independent of X_{ip} and has zero mean.

In econometrics, parametric models, such as linear or generalized linear models, and non-parametric models, such as spline regressions or generalized additive models, adopt their corresponding regression form. So, in simple models, instead of estimating the corresponding *P-dimensional* function $F(X_{ip})$, it is necessary only to obtain the *P+1* coefficient estimates β_p of the linear predictor $\beta_o + \beta_1 X_{i1} + \beta_1 X_{i2} + \cdots + \beta_p X_{ip}$.

Machine learning also uses alternative F in the form of classification and decision trees (Breiman et al., 1984), radial basis functions (Gomez-Verdejo et al., 2005), and random Markov fields (Dietterich et al., 2008), among others. The function F is known as a *base learner* in the machine learning literature.

Function F can be used to make inferences or predictions, or both. Even though econometric models are aimed at explanatory or predictive modelling, or both, non-econometric models are mainly used for prediction purposes (classification or regression problems),[3] because their F functions are not able to provide coefficient estimates that are directly interpretable as marginal effects.

When F is used for prediction purposes, given that (1) has an error term that averages zero, a predicted target variable $\hat{Y}_i$, for $\hat{F}$ that estimates the observed F, can be written as follows:

$$\hat{Y}_i = \hat{F}(X_{ip}). \tag{2}$$

In this setting, James et al. (2013) identify two types of errors: reducible and irreducible. When the expected value or average of the squared difference between the observed Y_i and predicted $\hat{Y}_i$ is taken, we obtain:

$$\mathrm{E}(Y_i - \hat{Y}_i)^2 = \mathrm{E}\left[F(X_{ip}) + \epsilon_i - \hat{F}(X_{ip})\right]^2, \tag{3}$$

which gives as a result:

$$\mathrm{E}(Y_i - \hat{Y}_i)^2 = \left[F(X_{ip}) - \hat{F}(X_{ip})\right]^2 + \mathrm{Var}(\varepsilon_i), \tag{4}$$

where the reducible error is $\left[F(X_{ip}) - \hat{F}(X_{ip})\right]^2$, and the irreducible error is $Var(\varepsilon_i)$ (variance of the error term). In fact, machine learning with non-econometric models aims to minimize the reducible error, which is equivalent to minimizing the distance between Y_i and $\hat{Y}_i$. This distance is known as the loss function, and will be denoted as $\varphi(Y_i, \hat{Y}_i)$.

Note that $Var(\varepsilon_i)$ cannot be reduced because these models only have a deterministic part that excessively learns from a given data set, in other words, they remove the only stochastic term. Consequently, highly accurate predictive machine learning algorithms such as certain tree-based or boosting-based techniques may result in overfitting, which means that the fitted models do not perform well on other databases. This is known as non-reproducibility. This result has also been verified by Pesantez-Narvaez et al. (2019).

Many loss functions have been proposed to develop machine learning algorithms with greater predictive accuracy. They must be convex and differentiable. This paper will focus on the exponential loss function that is used in a Logitboost:

[3] If Y_i is qualitative, we have with a classification problem, whereas if Y_i is quantitative, we have a regression problem. The latter must not be confused with a linear regression model. The machine learning and econometrics literatures have some discrepancies in terminology.

$$\varphi(Y_i\ ,\hat{Y}_i) = e^{Y_i \hat{Y}_i}. \quad (5)$$

In order to increase the predictive capacity, therefore, it makes sense to consider a simple econometric method like a base learner in a boosting-based algorithm. Firstly, the irreducible error may be effectively reduced by readjusting the base learner to improve the model fit. Secondly, the reducible error can also be computed. The statistical intuition behind choosing a primitive econometric model is that the newest iterations of boosting-based algorithms correct the prediction error by considering the previous iterations. This can be done more efficiently if the base learner is a weak[4] one, because there is more variability to learn in weak base learners than in strong ones that already have good predictive performance and no or almost no variability.

3. Description of Methodology

Three groups of boosting-based algorithms are considered: the classical econometric model, gradient boosting for classification and Logitboost-based algorithms. The first group consists of logistic regression. The second group consists of the original gradient boosting algorithm and gradient boosting tree. The third group consists of the original Logitboost and the proposed Synthetic Penalized Logitboost.

Note that $F(X_{ip}\ ; u)$ is the base learner mentioned earlier. It is a function of covariates X_{ip} and the parameters[5] represented by *u.*

In the data set that will be used in Section 4, there are n individuals and *P* covariates. The target variable Y_i is now an observed binary response variable that takes two values coded as 1 for the minority class (denied mortgage loan) and 0 for the majority class (approved for mortgage loan). Let *D* be the number of iterations of the boosting procedure, with *d=1,…,D*.

3.1 Logistic Regression

Let us assume that in the data set of *n* individuals and *P* covariates, the target variable Y_i is now an observed binary response variable that takes two values coded as 1 for the rare class and 0 for the majority class. A logistic regression is a classical econometric tool that is used to model and predict binary dependent variables explained by quantitative or qualitative covariates. It is a specific case of a generalized linear model when the link is the logit function and is given as:

$$\log \frac{P(Y_i=1)}{1-P(Y_i=1)} = \beta_o + \sum_{p=1}^{P} X_{ip}\beta_p\ , \quad (10)$$

where β_o, β_1, …, β_p are the model parameters, and $P(Y_i = 1)$ is the probability that Y_i equals 1 conditional on the covariates. By a simple algebraic manipulation, $P(Y_i = 1)$ is:

$$P(Y_i = 1)\ = E(Y_i)\ = \frac{e^{(\beta o + \sum_{p=1}^{P} X_{ip}\beta p)}}{1+e^{(\beta o+\sum_{p=1}^{P} X_{ip}\beta p)}}. \quad (11)$$

A logistic regression can be estimated by maximum likelihood method (for further details, see for example McCullagh and Nelder, 1989).

[4] Schapire and Freund (2013) define a weak learner as a particular case of base learner whose predictive performance is slightly better than chance, and typically far from zero.

[5] For example, if $F(X_{ip}\ ; u)$ is a regression model, u represents the coefficient estimates β, whereas if $F(X_{ip}\ ; u)$ is a classification and regression tree (CART), then u represents branches of the tree (splitting rules).

3.2 Gradient Boosting

The idea behind the Gradient Boosting proposed by Friedman (2001) is to compute a sum of optimized functions through an iterative process. The optimized functions are the result of a minimization of a loss function φ.

Let us assume that in the data set of *n* individuals and *P* covariates, the target variable Y_i is now continuous. The gradient boosting procedure starts with an initial guess of prediction $\hat{Y_i}^0$. It then consists of minimizing a loss function through an *argmin* between the observed $\hat{Y_i}$ and an arbitrary constant ρ.

$$\hat{Y_i}^0 = argmin_\rho \sum_{i=1}^{n} \varphi(Y_i, \rho). \tag{12}$$

Begin Algorithm:

For *d=1* to *D* do:

Let $\tilde{r_i}^d$ be the vector of the pseudo-residual which is the negative gradient of $\varphi(Y_i, \hat{Y_i}^d)$ at iteration *d.*

$$\tilde{r_i}^d = \frac{\partial \varphi(Y_i, \hat{Y_i}^d)}{\partial \hat{Y_i}^d} \mid_{Y_i = \hat{Y}_i^{d-1}}. \tag{13}$$

Then the squared error between the pseudo-residual and *F(X,u)* is minimized. This results in an updated u^d:

$$u^d = argmin_{u,\beta} \sum_{i=1}^{n} \left[\tilde{r_i}^d - \beta\, F\left(X_{ip}\, ; u^d\right)\right]^2. \tag{14}$$

Let γ be the result of a minimized loss function between the observed Y_i and $\hat{Y_i}^d$ + $\gamma\, F\left(X_{ip}\, ; u^d\right)$. Note that $\hat{Y_i}^d$ is the prediction from the given covariates X_{ip} and the updated parameters *u* at iteration *d*.

$$\gamma^d = argmin_\gamma \sum_{i=1}^{n} \varphi\left[Y_i, \hat{Y_i}^d + \gamma F\left(X_{ip}\, ; u^d\right)\right]. \tag{15}$$

The final prediction at iteration *D* is the sum of the previous prediction $\hat{Y_i}^{d-1}$ and $\gamma\hat{Y_i}^d$.

$$\hat{Y_i}^d = \hat{Y_i}^{d-1} + \gamma\, F\left(X_{ip}\, ; u^d\right). \tag{16}$$

End For
End Algorithm

3.3 Gradient L2 TreeBoost (Two-Class Logistic Boost)

Let us assume that in the data set of *n* individuals and *P* covariates, the target variable Y_i is now an observed binary response variable that takes two values coded as 1 for the rare class and 0 for the majority class. The L2 TreeBoost proposed by Friedman (2001) differs from the Original Gradient Boost in:

- Initial prediction $\hat{Y_i}^0$
- Loss function: Logistic loss function
- Base learner: Decision tree

The first estimation is calculated as follows:

$$\hat{Y}_i^{\,0} = \frac{1}{2} log \frac{1+\bar{Y}}{1-\bar{Y}}, \tag{17}$$

where $\bar{Y}$ is the mean of the dependent variable.

Begin Algorithm:

For *d=1* to *D* do: $$\tilde{r}_i^{\,d} = \frac{2\,Y_i}{1+\exp(2Y_i\hat{Y}_i^{\,d-1})} \tag{18}$$

The base learner $F(X_{ip}\,; u, R)$ equals $\sum_{j=1}^{J} u_j\, 1(X_{ip} \epsilon\, R_j)$ with J terminal nodes known as leaves, and R_j regions or classification rules, j=1,…, *J*. Parameters u correspond to the score of each leaf, which is the proportion of cases classified into Y_i given covariates X_{ip}. The tree-based algorithms are theoretically more efficient than linear or generalized linear methods in capturing non-linearities. The idea is that tree-based algorithms use information gain (measured by Gini impurity or entropy) to split a node. This helps to order the decision nodes associated with each covariate X_{ip}, so that the decision node with the highest information gain will split first, and so on until the one with lowest information gain. The information gain builds the R_j classification rules that map each observation *i* onto the correct leaf *j* by minimizing the entropy or Gini impurity of each node, so that the observations contained in the node are the most homogeneous (see further details in Hastie et al., 2009).

Now R_{jd} is computed by mapping all observations onto leaf *j* of tree (*j=1,…,J*) at iteration *d*, considering $\tilde{r}_i$ as the target variable and covariates X_{ip} as follows:

$$R_{jd} = j\text{- } leaf \text{ scores } (\tilde{r}_i\,, X_1^n). \tag{19}$$

Therefore γ_j^d is calculated for each leaf by minimizing a logistic loss function between the observed Y_i, and $\hat{Y}_i^{\,d-1} + \gamma^d$.

$$\gamma_j^d = argmin_\gamma \sum_{X_i\,\epsilon\, R_{jd}}^{n} \log\left[1 + \exp\left(-2Y_i\left(\hat{Y}_i^{\,d-1} + \gamma^d\right)\right)\right]. \tag{20}$$

However, since there is no closed form for the previous equation, an approximation of γ_j^d is obtained through the Newton-Raphson method as follows:

$$\gamma_j^d = argmin_\gamma \frac{\sum_{X_i\,\epsilon\, R_{jd}} \tilde{r}_i}{\sum_{X_i\,\epsilon\, R_{jd}} |\tilde{r}_i\,(2-|\tilde{r}_i|)|}. \tag{21}$$

And the final prediction $\hat{Y}_i^{\,d}$ is computed as:

$$\hat{Y}_i^{\,d} = \hat{Y}_i^{\,d-1} + \sum_{j=1}^{J} \gamma_j^d\; 1(X_i\,\epsilon\, R_{jd}). \tag{22}$$

End For
End Algorithm

Since tree-based algorithms generally overfit, decision tree pruning is considered in order to build a smaller tree with fewer *J* terminal nodes that lead to smaller variance by retaining the most relevant information and removing the least relevant (see further details in Hastie et al., 2009). For simplicity, Gradient L2 TreeBoost will be referred to as Gradient Tree Boost from here on.

3.4 Logitboost

The previous gradient boosting algorithms require the minimization of a loss function φ(Y_i , $\hat{Y}_i$). However, Friedman et al. (2000) have managed to approximate a logistic function as an additive logistic regression known as "Logitboost".

Let us assume that in the data set of *n* individuals and *P* covariates, the target variable Y_i is now an observed binary response variable that takes two values coded as 1 for the rare class and 0 for the majority class.

The Logitboost has some initial conditions:

- Initial prediction $\hat{Y}_i^{\,0}$ = 0.
- Let $p(X_i\,)$ be the probability estimates $p^0(X_i\,) = \frac{1}{2}$.

Begin Algorithm:

For *d = 1 to D* do:

This algorithm initializes by computing the working response z_i.

$$z_i^d = \frac{Y_i^{d-1} - p(X_i\,)^{d-1}}{p(X_i\,)^{d-1}(1-p(X_i\,)^{d-1})}. \tag{23}$$

In this case the $\boldsymbol{\chi}^2$ is a quadratic approximation of the log-likelihood with which a logistic regression can be estimated, as explained in Section 3.2. According to Friedman et al. (2000), the $\boldsymbol{\chi}^2$ can be a gentle alternative when the exponential loss function is used. Therefore, the working response z_i is an analogous expression to the pseudo-residuals $\tilde{r}_i$.

Again, the exponential loss function written in (5) is:

$$\varphi(Y_i\,, \hat{Y}_i) = e^{Y_i\hat{Y}_i}.$$

$$e^{Y_i\hat{Y}_i} = \frac{|Y_i - p(X_i\,)|}{\sqrt{p(X_i\,)(1-p(X_i\,))}}, \tag{24}$$

where $\hat{Y}_i$ is obtained as follows:

$$\hat{Y}_i^{\,d} = \frac{1}{2} log \frac{p(X_i\,)^{d-1}}{(1-p(X_i\,)^{d-1})}. \tag{25}$$

A vector of weights w_i is computed as follows:

$$w_i^{\,d} = p(X_i\,)^{d-1}\,(1 - p(X_i\,)^{d-1}). \tag{26}$$

A base learner $F(X_i, u)$ must be trained by fitting a weighted least squares regression as explained in Section 3.1, with a vector of weights w_i and a target variable z_i. Note that even though a binary target variable is set for this boosting, this *F* admits continuous target variables. The reason is that the working response z_i transforms the binary variable Y_i into a continuous one, so that two classes are still found in the first iteration. However, from the second iteration onwards, observations of z_i start to change during the boosting, so that at the end several values of z_i are found.

$$\beta^d = argmin_\beta \sum_{i=1}^{n} w_i \left[z_i^d - \left(\beta_o + \sum_{p=1}^{P} X_{ip}\beta_p\right)\right]^2 . \tag{27}$$

$\hat{Y}_i^{\,d}$ has to be updated as follows:

$$\hat{Y}_i^{\,d} = \hat{Y}_i^{\,d-1} + \frac{1}{2} F\left(X_{ip}\,; u^d\right). \tag{28}$$

Parameters u^d are the coefficient estimates β obtained in the linear regression.

Then the probabilities have to be updated:

$$p(X_i\,)^d = \frac{\exp\left(\hat{Y}_i^{\,d}\right)}{\exp\left(\hat{Y}_i^{\,d}\right) + \exp\left(-\hat{Y}_i^{\,d}\right)}. \tag{29}$$

End For
End Algorithm

3.5 Synthetic Penalized Logitboost

The proposed Synthetic Penalized Logitboost incorporates slight changes to the original Logitboost and introduces a new alternative weighting mechanism w_i. This methodological proposal was particularly motivated by Pesantez-Narvaez and Guillen (2020a, 2020b). They managed to propose weighting corrections in parametric models to improve their predictive performance for binary dependent variables.

We keep the two initial conditions for $\hat{Y}_i^{\,0}$ and $p^0(X_i\,)$:

- Initial prediction: $\hat{Y}_i^{\,0} = 0$.
- Let $p(X_i\,)$ be the probability estimates $p^0(X_i\,) = \frac{1}{2}$.

Begin Algorithm:

For *d = 1 to D* do:

This algorithm initializes by computing the working response z_i.

$$z_i^d = \frac{Y_i^{\,d-1} - p(X_i\,)^{d-1}}{p(X_i\,)^{d-1}\left(1 - p(X_i\,)^{d-1}\right) + \delta}. \tag{30}$$

where δ is a very small number (close to zero), e.g. 0.0001, so we avoid division by zero.

$\hat{Y}_i$ is obtained as follows:

$$\hat{Y}_i^{\,d} = \frac{1}{2} log \frac{p(X_i\,)^{d-1}}{\left(1 - p(X_i\,)^{d-1}\right)}. \tag{31}$$

A vector of weights w_i is computed as follows:

$$w_i^{\,d} = \begin{cases} p(X_i\,)^{d-1}\left(1 - p(X_i\,)^{d-1}\right) + \bar{Y}\left|Y_i - p(X_i\,)^{d-1}\right| & if\ |Y_i\ - p(X_i\,)| < \bar{Y} \\ p(X_i\,)^{d-1}\left(1 - p(X_i\,)^{d-1}\right) & if\ |Y_i\ - p(X_i\,)| \geq \bar{Y} \end{cases} \tag{32}$$

This weighting mechanism aims to penalize by giving less weight to observations whose distance between the observed Y_i and the probability estimates $p(X_i\,)$ is greater than the mean of the dependent variable. In other words, we penalize observations which are more likely to be misclassified. This weighting mechanism leads to stabilization after very few iterations of the boosting procedure.

Weights must be normalized by dividing by the sum of the vector of weights:

$$w_i^{\,d} = \frac{w_i^{\,d}}{\sum_{i=1}^{n} w_i^{\,d}}. \tag{33}$$

$F(X_{ip}\,;u)$ has to be trained as weighted least squares with weights w_i:

$$\beta^d = argmin_\beta \sum_{i=1}^{n} w_i \left[z_i^d - \left(\beta_o + \sum_{p=1}^{P} X_{ip}\beta_p\right)\right]^2 . \tag{34}$$

$\hat{Y_i}^d$ has to be computed as follows:

$$\hat{Y_i}^d = \hat{Y_i}^{d-1} + \frac{1}{2}F(X_{ip}\,;u^d). \tag{35}$$

And we must update the probabilities:

$$p(X_i)^d = \min\left\{\frac{1}{1+\exp\left(-2\hat{Y_i}^{d-1}\right)} + \delta\,,1\right\}. \tag{36}$$

The final $p(X_i)$ is related to the log-odds through (35).

$$p^d(Y_i = 1|X) = \frac{1}{1+\exp\left(-2\hat{Y_i}^{d-1}\right)}. \tag{37}$$

$$p^d(Y_i = 0|X) = \frac{1}{1+\exp\left(2\hat{Y_i}^{d-1}\right)}. \tag{38}$$

End For
End Algorithm

4. Illustrative Data and Descriptive Statistics

Table 1. Description of the Home Mortgage Disclosure Act (HDMA) cross-section data set.

Variables	Description
Dir	debt payment to total income ratio.
Hir	housing expenses to income ratio.
Lvr	ratio of size of loan to assessed value of property.
Css	consumer credit score from 1, as the best score, to 6 as the lowest score.
Mcs	mortgage credit score from 1, as the best score, to 4 as the lowest score.
Uria	1989 Massachusetts unemployment rate in the applicant's industry.
Pbcr	whether the applicant has a public bad credit record.
Dmi	whether the applicant was denied mortgage insurance.
Self	whether the applicant is self-employed.
Single	whether the applicant is single.
Condominium	whether the applicant lives in a condominium.
Black	whether the applicant is black.
Y	which was coded as 1 when the mortgage application was denied, and 0 otherwise.

In order to illustrate the proposed methodology, we use a publicly available Home Mortgage Disclosure Act (HDMA) cross-section data set, which was collected by the U.S. Government through a survey designed to gather additional information on minority group applicants. The intention was to uncover whether discrimination based on the applicants' race occurs in mortgage lending. The sample has 2381 applicants who were chosen by a simple random sample in Boston, Massachusetts (United States) in 1997-1998.[6] There is an equal number of denials among white and minority applicants in order to provide sufficient power to validate any discrimination. The HDMA database is also available in the *Ecdat* package in R. We drop the last observation due to missingness; no imputation technique was necessary.

Table 1 describes the variables in the Home Mortgage Disclosure Act (HDMA) cross-section data set.

Table 2. Descriptive statistics for the HDMA data set (1997-1998). Mean of continuous covariates in the denied group, in the approved group and in the total. Counts and row proportions are shown for dichotomous covariates.

Variables		**Denied Mortgage Application ($Y_i = 1$)**	**Approved Mortgage Application ($Y_i = 0$)**	**Total**
Dir		0.389	0.323	0.331
Hir		0.290	0.251	0.255
Lvr		0.816	0.727	0.738
Css		3.302	1.955	2.116
Mcs		1.881	1.699	1.721
Uria		4.014	3.742	3.774
Pbcr	No	209 (9.48%)	1996 (90.52%)	2205
	Yes	76 (43.43%)	99.0 (56.57%)	175
Dmi	No	241 (10.33%)	2091 (89.67%)	2332
	Yes	44 (91.67%)	4 (8.33%)	48
Self	No	239 (11.36%)	1864 (88.64%)	2103
	Yes	46 (16.61%)	231 (83.39%)	277
Single	No	144 (9.97%)	1300 (90.03%)	1444
	Yes	141 (15.06%)	795 (84.94%)	936
Condominium	No	189 (11.16%)	1505 (88.84%)	1694
	Yes	96 (13.99%)	590 (86.01%)	686
Black	No	189 (9.26%)	1852 (90.74%)	2041
	Yes	96 (28.32%)	243 (71.68%)	339
Total		285 (11.97%)	2095 (88.03%)	2380

Source: Authors' own calculations based on HDMA data (1997-1998). The variables refer to the debt payment to total income ratio (Dir); housing expenses to income ratio (Hir); ratio of size of loan to assessed value of property (Lvr); consumer credit score from 1, as the best score, to 6 as the lowest score (Css); mortgage credit score from 1, as the best score, to 4 as the lowest score (Mcs); whether the applicant has a public bad credit record (Pbcr); whether the applicant was denied mortgage insurance (Dmi); whether the applicant is self-employed (Self); whether the applicant is single (Single); 1989 Massachusetts unemployment rate in the applicant's industry (Uria); whether the applicant lives in a condominium (Condominium); whether the applicant is black (Black); and finally, the mortgage application (Y), which was coded as 1 when the mortgage application was denied, and 0 otherwise.

Table 2 above shows the descriptive statistics for the HDMA data set. The last row reveals that a substantial part of the sample has an approved mortgage application (88.03%). The mean ratios corresponding to the debt to total income and housing expenses to income are slightly higher for applicants whose mortgage application was denied, which means that their debt is higher than it is for the other applicants. Additionally, the mean ratio of the size of loan to assessed value of property is almost 9% higher for people with a denied mortgage application.

[6] Even though these data are old, we believe that they are useful to show the implementation and testing of the newly proposed model since the data set contains the required variables to replicate the model proposed by Munnell et al. (1996).

The credit score and mortgage score of approved mortgage applicants are, respectively, 0.6 times and 0.88 better than the scores of denied applicants. Whereas 56.57% of applicants with a bad public credit record were approved, 43.43% were denied. Moreover, 8.33% of applicants who were denied mortgage insurance had an approved mortgage application, while 91.67% were also denied their mortgage application. While 83.39% of self-employed applicants were approved, 88.65% of applicants who were not self-employed were approved. Also, 84.94% of single applicants were approved, while 15.06% were not. There is a slight percentage difference between applicants who live in a condominium and had an approved mortgage application and applicants who live in a condominium and had a denied mortgage application. Lastly, 71.68% of black applicants were approved, while 90.74% of non-black applicants were approved.

5. Results and Discussion

This section contains two parts. The first part presents the results of the prediction performance of the Synthetic Penalized Logitboost in comparison to the algorithms described in Section 3. The results are shown below based on three calculations. The second part presents a proposal to recover the interpretability of the Synthetic Penalized Logitboost model.

5.1 Prediction Performance

Table 3 presents the root-mean-square error[7] (RMSE) of Logistic regression, Logitboost, Gradient Tree Boost and the Synthetic Logitboost, tested for three scenarios: the entire sample (all observations), the observations that correspond to Y_i=1, and the observations that correspond to Y_i=0. The RMSE is suitable to measure the distance between the observed Y_i and the predicted $\hat{Y}_i$, so the predictive performance will not depend for example on the precision of the threshold picked to build a confusion matrix.

The Gradient Boost (tree) is built with the model developer's default hyperparameters from the *gbm* package in R, which correspond to the number of trees (100), the maximum depth of variable interactions (1), the minimum number of observations in the terminal nodes of the trees (10), and shrinkage (0.1). The Gradient Boost (tree) GS-CV is built with 10-fold cross validation and optimized hyperparameters through grid search, which correspond to the number of trees (150), the maximum depth of variable interactions (2), the minimum number of observations in the terminal nodes of the trees (10), and shrinkage (0.1) with the *caret* package in R. Logistic, Logitboost, and Synthetic Penalized Logitboost are built according to the definitions in Section 3, and they do not have hyperparameters.

In the first calculation, Logistic regression and Logitboost perform almost the same, confirming numerically what was noted theoretically. Synthetic Penalized Logitboost has a smaller RMSE in some of the lowest and highest accumulated predictions, even when it is compared with the Gradient Tree Boosting models (with and without optimized hyperparameters).

When analysing the observations that correspond to denied applications (Y_i=1), both Gradient Tree Boost models perform worse than Logistic and Logitboost for some high score predictions. This confirms the fact that optimized Gradient Tree Boost methods risk failing to predict the minority class (Y_i=1) even when their performance is better with the complete data set. However, the Synthetic Penalized Logitboost performs better than Logistic and Logitboost in the lowest accumulated predictions, and better than the Gradient Tree Boost GS-CV in the 1% and 5% highest accumulated predictions.

[7] The root-mean-square error is calculated as follows: $\sqrt{\sum_{i=1}^{n} \frac{(Y_i - \hat{Y}_i)^2}{n}}$.

Table 3. Root-Mean-Square Error of Logistic regression, Logitboost, Gradient Tree Boost and the Synthetic Penalized Logitboost, tested for the entire sample, when Y_i=1, and when Y_i=0.

	RMSE (All Observed Y_i)											
Methods	**Lower Extreme**						**Upper Extreme**					
	0.01	**0.05**	**0.10**	**0.20**	**0.30**	**0.40**	**0.01**	**0.05**	**0.10**	**0.20**	**0.30**	**0.40**
Logistic	0.0067	0.2025	0.1696	0.1635	0.1614	0.1605	0.2739	0.4143	0.4510	0.4236	0.3884	0.3661
Logitboost	0.0064	0.2026	0.1697	0.1635	0.1614	0.1575	0.2739	0.4143	0.4511	0.4236	0.3884	0.3649
Gradient Tree Boost	0.0266	0.0931	0.1566	0.1438	0.1439	0.1687	0.1975	0.4020	0.4595	0.4351	0.3905	0.3583
Gradient Boost (tree) GS - CV	0.0182	0.0195	0.1432	0.1362	0.1236	0.1468	0.1918	0.3391	0.4235	0.4083	0.3698	0.3426
Synthetic Penalized Logitboost	0.0094	0.1568	0.1568	0.1631	0.1610	0.1540	0.2711	0.4138	0.4569	0.4297	0.3890	0.3678
	RMSE (When Y_i= 1)											
Methods	**Lower Extreme**						**Upper Extreme**					
	0.01	**0.05**	**0.10**	**0.20**	**0.30**	**0.40**	**0.01**	**0.05**	**0.10**	**0.20**	**0.30**	**0.40**
Logistic	0.9875	0.9802	0.9693	0.9527	0.9368	0.9173	0.0051	0.0110	0.0402	0.1475	0.2673	0.3625
Logitboost	0.9879	0.9808	0.9701	0.9536	0.9379	0.9185	0.0051	0.0108	0.0404	0.1474	0.2673	0.3626
Gradient Tree Boost	0.9706	0.9663	0.9616	0.9503	0.9304	0.9063	0.0150	0.0413	0.0747	0.1661	0.2771	0.3605
Gradient Boost (tree) GS - CV	0.9779	0.9732	0.9671	0.9517	0.9273	0.8930	0.0112	0.0321	0.0498	0.0898	0.1613	0.2441
Synthetic Penalized Logitboost	0.9836	0.9775	0.9684	0.9540	0.9393	0.9210	0.0061	0.0147	0.0502	0.1526	0.2777	0.3770
	RMSE (When Y_i= 0)											
Methods	**Lower Extreme**						**Upper Extreme**					
	0.01	**0.05**	**0.10**	**0.20**	**0.30**	**0.40**	**0.01**	**0.05**	**0.10**	**0.20**	**0.30**	**0.40**
Logistic	0.0064	0.0113	0.0151	0.0205	0.0251	0.0296	0.7233	0.4759	0.3694	0.2797	0.2352	0.2069
Logitboost	0.0061	0.0109	0.0146	0.0198	0.0244	0.0289	0.1210	0.0889	0.0739	0.0584	0.0490	0.0426
Gradient Tree Boost	0.0266	0.0270	0.0285	0.0310	0.0331	0.0353	0.6737	0.4504	0.3521	0.2665	0.2240	0.1968
Gradient Boost (tree) GS - CV	0.0180	0.0194	0.0207	0.0231	0.0251	0.0272	0.7044	0.4596	0.3545	0.2651	0.2209	0.1935
Synthetic Penalized Logitboost	0.0092	0.0136	0.0168	0.0214	0.0252	0.0290	0.7055	0.4731	0.3676	0.2777	0.2334	0.2051

Three calculations are presented above. The first set of results is displayed in the top part of Table 3 (All Observed Y), where RMSE is calculated for all observations in the sample. The second set of results is displayed in the middle of Table 3 (When Y=1) only for observations that correspond to denied applications, whereas the third set of results is displayed in the bottom part of Table 3 (When Y=0) only for observations that correspond to approved applications. All results are analysed by groups of scores. So, each RMSE for 1%, 5%, 10%, 20%, 30% and 40% of the lowest accumulated prediction scores is shown on the left-hand side of the table under "Lower Extreme", and each RMSE for 1%, 5%, 10%, 20%, 30% and 40% of the highest accumulated prediction scores is shown on the right-hand side of the table under "Upper Extreme".

When analysing the observations that correspond to accepted applications (Y_i=0), Logitboost differs considerably from Logistic in the highest predictions, where it performs much better, while in the lowest scores, the results are very similar for both models. Now, Gradient Tree Boost GS-CV performs better than the two classical methods, while Synthetic Penalized Logitboost also generally performs better than the classical methods.

Table 4. Root-Mean-Square Error of Logistic regression, Logitboost, Gradient Tree Boost and the Synthetic Penalized Logitboost for the training and testing HMDA data sets.

RMSE for Testing Data Set

Methods	**Lower Extreme**						**Upper Extreme**					
	0.01	**0.05**	**0.10**	**0.20**	**0.3**	**0.4**	**0.01**	**0.05**	**0.10**	**0.20**	**0.3**	**0.4**
Logistic	0.0059	0.0091	0.0113	0.2582	0.2730	0.2579	0.1650	0.4374	0.4276	0.4239	0.3874	0.3645
Logitboost	0.0069	0.1812	0.1274	0.1279	0.1051	0.1275	0.0000	0.4006	0.4449	0.4051	0.3960	0.3602
Gradient Boost (tree)	0.4880	0.2312	0.1648	0.1647	0.1647	0.2150	0.0420	0.4361	0.4618	0.4563	0.4151	0.3702
Gradient Boost (tree) GS - CV	0.0114	0.0135	0.1656	0.1841	0.1781	0.1838	0.0477	0.3956	0.4519	0.4179	0.3966	0.3659
Synthetic Penalized Logitboost	0.0064	0.0078	0.1272	0.0906	0.1041	0.1270	0.0001	0.4354	0.4369	0.4149	0.4079	0.3750

RMSE for Training Data Set

Methods	**Lower Extreme**						**Upper Extreme**					
	0.01	**0.05**	**0.10**	**0.20**	**0.3**	**0.4**	**0.01**	**0.05**	**0.10**	**0.20**	**0.3**	**0.4**
Logistic	0.0054	0.1982	0.1981	0.1975	0.1620	0.1412	0.3856	0.4017	0.4472	0.4136	0.3658	0.3444
Logitboost	0.0070	0.2033	0.2190	0.1943	0.1910	0.1851	0.2541	0.3964	0.4499	0.4245	0.3892	0.3645
Gradient Boost (tree)	0.2825	0.1798	0.1565	0.1680	0.1565	0.1790	0.0512	0.3711	0.4449	0.4274	0.3847	0.3698
Gradient Boost (tree) GS - CV	0.0100	0.0124	0.0146	0.0183	0.0783	0.0953	0.0516	0.2964	0.4148	0.4082	0.3738	0.3451
Synthetic Penalized Logitboost	0.0055	0.0084	0.0110	0.1213	0.1587	0.1618	0.0205	0.4362	0.4594	0.4364	0.3991	0.3792

The HMDA database was randomly split into training data (70%) and testing data (30%). Each RMSE for 1%, 5%, 10%, 20%, 30% and 40% of the lowest accumulated prediction scores is shown on the left-hand side of the table under "Lower Extreme", and each RMSE for 1%, 5%, 10%, 20%, 30% and 40% of the highest accumulated prediction scores is shown on the right-hand side of the table under "Upper Extreme".

It can be concluded that Synthetic Penalized Logitboost makes slightly more accurate predictions than the other algorithms in most observations for the scores in the upper and lower extremes.

The second calculation in Table 4 shows the RMSE of the previously discussed methods split into testing and training HMDA data sets. The Synthetic Penalized Logitboost performs quite similarly in the training and testing data sets. This result might be explained by the fact that the algorithm is built with an error term that allows for random variation in covariates when modelling the target variable; and consequently, it avoids overfitting. A similar behaviour is obtained with logistic regression, which is a parametric model. Gradient Tree Boost requires hyperparameter optimization and cross-validation procedures to correct overfitting.

While correction methods to avoid overfitting are widely accepted in the machine learning literature, it is risky in terms of interpretation to tune shrinkage parameters. As their values increase, they deliberately shrink or disappear variables (nodes) with smaller entropy or Gini impurity. However, empirical econometric analysis demands the measurement of the coefficient estimates even when they are not significant in the model; otherwise the analyst may lose control of their natural effect on the dependent variable.

The third calculation in Table 5 presents the predictive measures of the discussed methods. The Synthetic Penalized Logitboost has more accuracy than Logistic and Logitboost and more specificity than Gradient Boost (Tree) GS-CV in the testing data sets. In aggregate terms, the Synthetic Penalized Logitboost has larger RMSE than alternative methods. Note that the error correction through penalization is focused on observations which are far from the average values, so the proposed method tends not to affect the predictive improvement of mean observations.

Table 5. Predictive measures of Logistic regression, Logitboost, Gradient Tree Boost and the Synthetic Penalized Logitboost for the testing and training HMDA data sets.

Testing Data Set				
Predictive Measures	**Logistic Regression**	**Logitboost**	**Gradient Boost (Tree) GS - CV**	**Synthetic Penalized Logitboost**
Recall	0.7000	0.7333	0.7333	0.6556
Specificity	0.7837	0.8269	0.8381	0.8446
Accuracy	0.7731	0.8151	0.8249	0.8207
Precision	0.3182	0.3793	0.3952	0.3782
F1 Score	0.4375	0.5000	0.5136	0.4797
RMSE	0.2774	0.2654	0.2669	0.3327
Training Data Set				
Predictive Measures	**Logistic Regression**	**Logitboost**	**Gradient Boost (Tree) GS - CV**	**Synthetic Penalized Logitboost**
Recall	0.7026	0.7026	0.7282	0.6872
Specificity	0.809	0.811	0.8525	0.7967
Accuracy	0.7965	0.7983	0.8379	0.7839
Precision	0.3278	0.3301	0.3955	0.3095
F1 Score	0.447	0.4492	0.5126	0.4268
RMSE	0.2757	0.2757	0.259	0.2780

The HMDA database was randomly split into training data (70%) and testing data (30%). The threshold used to convert the continuous response into a binary response is the mean of the outcome variable. Recall measures the ratio of applicants who were classified in the denied mortgage application group to those who were effectively denied. Specificity measures the ratio of applicants who were classified in the denied group to those who were not denied. Accuracy measures the proportion of applicants who are correctly classified. Precision is the ratio of correctly predicted denied applicants to the total predicted denied applicants. The F1 Score is the weighted average of Precision and Recall.

We observe quite similar patterns of performance when the Synthetic Penalized Logitboost is applied to data sets that have low frequencies, for example, in HDMA 2012 and 2017. The results obtained for the testing and training data sets are very close to each other and do not differ significantly. Moreover, the Synthetic Penalized Logitboost has lower RMSE than the alternative methods in the 1% and/or 5% lower and upper extremes. Further details and discussion of results obtained with HDMA 2012 and 2017 are presented in the Appendix.

Figure 1 shows the evolution of the RMSE within 100 iterations of the Synthetic Penalized Logitboost. This algorithm gets the RMSE stable after many iterations. While there is no theoretical guarantee that the proposed method will stabilize after some iterations, we obtained similar behaviour when applying the Synthetic Penalized Logitboost to the HDMA 2012 and 2017 data sets. We propose trying alternative initial values if this does not happen.

The RMSE is smaller and more homogeneous for observations in the minority group (Y_i=1) in the lowest predictions, while the RMSE is larger and more heterogenous for observations in the majority group (Y_i=0) in the highest predictions. In aggregate terms, the lowest 1% and the highest 1% of predicted scores (extreme values) have a much more accurate performance than the other accumulated percentages of predictions.

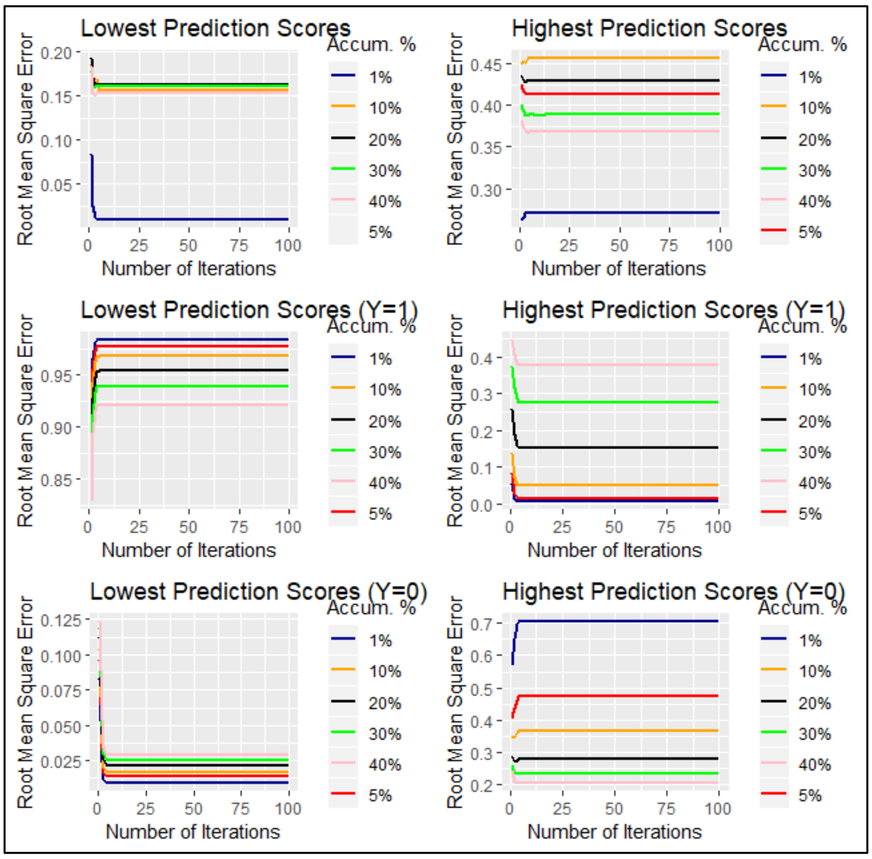

Figure 1. RMSE data set across 100 iterations of the Synthetic Penalized Logitboost for the HDMA data set.

5.2 Recovering the interpretability of the model

Machine learning algorithms are sometimes considered black boxes since their interpretability is not straightforward. In contrast, the Synthetic Penalized Logitboost can be seen as a method that recalibrates a least square regression in reweighted versions and penalizes incorrect predictions, so its interpretability can be recovered. Let us note again in Figure 1 that when the RMSE achieves stabilization in the boosting procedure (minimum variance), so too do the coefficient estimates of the model. Therefore, if the coefficients are averaged, one might gain some intuition about the sign and magnitude of the covariate effect on the response.

Table 6 shows the coefficient estimates obtained by a logistic regression and the Synthetic Penalized Logitboost. The results obtained by the logistic regression are consistent with the conclusions obtained by Munnell et al. (1996).
Moreover, the sign of the mean of the coefficient estimates of the Synthetic Penalized Logitboost within iterations is almost the same before and after the stabilization. The signs and the magnitude of the coefficients are consistent with the ones obtained by logistic regression. Nonetheless, the magnitude seems to be expressed on another scale, which was expected since the target variable used in the two methods is not the same.

Regarding the economic interpretation, Table 6 provides interesting results. Both the applicants with a high debt payment to income ratio and the applicants with a high ratio of size of loan to assessed value of property are more likely to receive a denied mortgage application. Moreover, the applicants with the lowest consumer and mortgage credit scores are more likely to be denied. Single applicants are more likely than non-single applicants to have a denied mortgage application. A higher unemployment rate in the applicant’s industry is also more likely to result in denial. Last but not least, black applicants are more likely than others to have a denied mortgage application, even when controlling for all the ratios and factors included in the model. The Synthetic Penalized Logitboost provides similar interpretations, as the mean coefficients have almost the same sign[8] as the logistic regression coefficients, even though they are not directly comparable in size.

Table 6. Coefficient Estimates for the Logistic Regression and the Synthetic Penalized Logitboost in the HDMA data set.

Coefficient Estimates	**Logistic Regression**			**Synthetic Penalized Logitboost**			
	Lower Bound	**Estimate**	**Upper Bound**	**Minimum**	**Mean**	**Maximum**	**Mean (After Stabilization)**
Intercept *	-8.2496	-7.1289	-6.0610	-3.0488	-0.0641	0.0027	-0.0037
Dir *	2.7441	4.7742	6.8259	-0.0196	0.0430	1.8492	0.0016
Hir	-2.8311	-0.4221	2.0323	-0.4991	-0.0166	0.0036	-0.0043
Lvr *	0.8324	1.7980	2.7881	-0.0086	0.0122	0.4326	0.0009
Css *	0.2168	0.2948	0.3726	0.0000	0.0031	0.1243	0.0003
Mcs *	3.5380	4.5154	5.7623	-0.0008	0.0027	0.0751	0.0005
Pbcr	-0.0334	0.2464	0.5243	-0.0153	0.0122	0.8070	0.0000
Dmi *	0.8239	1.2281	1.6259	-0.0066	0.0449	2.8376	0.0013
Self *	0.1972	0.6224	1.0305	-0.0003	0.0066	0.2217	0.0007
Single *	0.1015	0.4078	0.7141	-0.0009	0.0055	0.1524	0.0011
Uria *	0.0002	0.0687	0.1336	-0.0001	0.0007	0.0232	0.0001
Condominium	-0.3677	-0.0320	0.2970	-0.0142	-0.0002	0.0053	0.0001
Black *	0.3707	0.7266	1.0753	-0.0002	0.0081	0.3526	0.0006

The logistic regression columns show the point estimates of the lower and upper bounds of a 95% confidence interval. The XGBoost columns show the means of the coefficient estimates with a linear boosting of the D iterations. Similarly, the bounds are presented with the minimum and maximum values in the iterations. The stabilization starts from the fourth iteration onwards. * indicates that the coefficient is significant at the 90% confidence level in the logistic regression estimation. The calculations were performed in R and scripts are available from the authors.

6. Conclusions

We borrowed the mortgage lending model specification put forward by Munnell et al. (1996) to provide a real-life application in empirical economics using the proposed algorithm. We conclude that weighting corrections in machine learning algorithms with an econometric base learner can improve the predictive performance by decreasing the RMSE in several segments of the predictions. The Synthetic Penalized Logitboost preserves a stochastic term and trains a weighted linear regression as base learner in order to prevent overfitting. Hence, the algorithm can be used to reproduce alternative data sets without losing power.

Although the improvement in predictive performance is not excessively high, we provide evidence that it can lead to smaller RMSE than the Gradient Tree Boost (recognized for smartly capturing non-linearities) for observations that belong to the minority class in imbalanced data problems that tend to be underestimated by econometric methods and machine learning algorithms in general.

Beyond that, empirical sciences face challenges with machine learning architecture when their purpose is not only to make predictions using imbalanced data, but also to explain their causes

[8] Note that the mean of the coefficient estimates of *Pbcr* and *Condominium* differ from the logistic regression. However, the effect of *Pbcr* is similar to the findings of Munnell et al. (1996), and the effect of *Condominium* should be analysed in depth since more types of living spaces (more and less expensive) must be controlled for to verify payment guarantee.

in detail. On one hand, economists have used econometrics thus far to analyse the determinants of a specific phenomenon, but some models tend to be simplified due to the rigidity of linear specifications in most classical models. On the other hand, machine learning handles more large-scale complex data accurately but cannot provide direct coefficient estimates to link the corresponding effects of exogenous variables on the response outcome. The Synthetic Penalized Logitboost has started to combine these two approaches by providing some statistical intuition of its coefficient estimates since the base learner is a weighted least squares regression. As a result, the model always stabilizes its coefficients, while also being able to deal with complex structures and imbalanced phenomena.

Since the Synthetic Penalized Logitboost strongly penalizes observations whose probability estimates deviate considerably from the observed target variable, we wonder whether the predictive performance could be further improved in more imbalanced data sets or more complex models than the one presented here. While the model specification in Munnell et al. (1996) works with tailor-made survey data, our proposed model can also work with extensive data obtained through web scraping or with device-collected data.

Acknowledgements: We would like to thank the Spanish Ministry of Economy, FEDER grant ECO2016-76203-C2-2-P, Fundación BBVA, and ICREA Academia.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix

This section provides the results of the prediction performance of the Synthetic Penalized Logitboost in comparison to the algorithms described in Section 3 for the HDMA 2012 and HDMA 2017 datasets.

Table 7 shows the RMSE of Logistic regression, Logitboost, Gradient Tree Boost and Synthetic Penalized Logitboost for the training and testing HDMA 2012 data sets. The Synthetic Penalized Logitboost has a lower RMSE than the other methods, especially in the 1% and 5% of lower and upper extremes. The second-best prediction performance for the lower extremes corresponds to the results obtained by the Logistic and Logitboost with a prediction error equal to zero, while second-best for the upper extremes corresponds to the Gradient Boost (tree) GS-CV. Moreover, the Synthetic Penalized Logitboost has a similar performance in the testing and training data sets.

Table 8 presents additional predictive measures of Logistic regression, Logitboost, Gradient Tree Boost and the Synthetic Penalized Logitboost for the testing and training HDMA 2012 data sets. The Synthetic Penalized Logitboost has the highest recall with similar rates in the training and testing data sets. The second highest recall corresponds to the Gradient Boost (tree) GS – CV. Figure 2 shows RMSE across 100 iterations of the Synthetic Penalized Logitboost for the HDMA 2012. Approximately the first 5 to 10 iterations have brusque changes, however after iteration 30 approximately the RMSE gets stable.

Table 9 shows the RMSE of Logistic regression, Logitboost, Gradient Tree Boost and Synthetic Penalized Logitboost for the training and testing HDMA 2017 data sets. All methods have a prediction error equal to zero in the lower extreme, while the Penalized Logitboost and Logistic regression have the smallest RMSE in the upper extremes. Additionally, Table 10 presents alternative predictive measures for the mentioned methods. The Synthetic Penalized Logitboost has again the highest recall, even when RMSE in aggregated terms is higher than others. Finally, Figure 3 shows that the RMSE gests stable after iteration 40 approximately.

Table 7. Root-Mean-Square Error of Logistic regression, Logitboost, Gradient Tree Boost and the Synthetic Penalized Logitboost for the training and testing HMDA 2012 data sets.

RMSE for Testing Data Set

Methods	Lower Extreme						Upper Extreme					
	0.01	0.05	0.10	0.20	0.3	0.4	0.01	0.05	0.10	0.20	0.3	0.4
Logistic	0	0	0	0	0	0	0.4941	0.4890	0.4762	0.4632	0.4518	0.4414
Logitboost	0	0	0	0	0	0	0.4950	0.4896	0.4774	0.4634	0.4521	0.4407
Gradient Boost (tree)	0.0008	0.0011	0.0025	0.0033	0.0038	0.0041	0.5081	0.4882	0.4772	0.4605	0.4494	0.4386
Gradient Boost (tree) GS - CV	0.0461	0.0239	0.0172	0.0155	0.0126	0.0110	0.4788	0.4774	0.4713	0.4599	0.4510	0.4406
Synthetic Penalized Logitboost	0	0	0	0	0	0	0.4730	0.4619	0.4639	0.4553	0.4476	0.4429

RMSE for Training Data Set

Methods	Lower Extreme						Upper Extreme					
	0.01	0.05	0.10	0.20	0.3	0.4	0.01	0.05	0.10	0.20	0.3	0.4
Logistic	0	0	0	0	0	0	0.4947	0.4886	0.4773	0.4639	0.4519	0.4403
Logitboost	0.0150	0.0067	0.0067	0.0058	0.0048	0.0041	0.4956	0.4892	0.4779	0.4639	0.4522	0.4407
Gradient Boost (tree)	0.0010	0.0012	0.0029	0.0038	0.0043	0.0046	0.5040	0.4890	0.4774	0.4631	0.4511	0.4399
Gradient Boost (tree) GS - CV	0.0377	0.0218	0.0167	0.0157	0.0129	0.0123	0.4779	0.4768	0.4716	0.4603	0.4513	0.4404
Synthetic Penalized Logitboost	0	0	0	0	0	0	0.5001	0.4853	0.4759	0.4629	0.4507	0.4413

The HMDA database was randomly split into training data (70%) and testing data (30%). Each RMSE for 1%, 5%, 10%, 20%, 30% and 40% of the lowest accumulated prediction scores is shown on the left-hand side of the table under "Lower Extreme", and each RMSE for 1%, 5%, 10%, 20%, 30% and 40% of the highest accumulated prediction scores is shown on the right-hand side of the table under "Upper Extreme". The Gradient Boost (tree) GS – CV is built with 10-fold cross validation and optimized hyperparameters through grid search.

Table 8. Predictive measures of Logistic regression, Logitboost, Gradient Tree Boost and the Synthetic Penalized Logitboost for the testing and training HMDA 2012 data sets.

Testing Data Set

Predictive Measures	Logistic Regression	Logitboost	Gradient Boost (Tree) GS - CV	Synthetic Penalized Logitboost
Recall	0.9946	0.9954	0.9978	0.9981
Specificity	0.6552	0.6548	0.6532	0.6511
Accuracy	0.6939	0.6935	0.6923	0.6905
Precision	0.271	0.2698	0.2693	0.2682
F1 Score	0.4259	0.4245	0.4241	0.4228
RMSE	0.2851	0.2845	0.2842	0.2877

Training Data Set

Predictive Measures	Logistic Regression	Logitboost	Gradient Boost (Tree) GS - CV	Synthetic Penalized Logitboost
Recall	0.9955	0.9963	0.998	0.9983
Specificity	0.6548	0.6541	0.653	0.651
Accuracy	0.6935	0.6928	0.692	0.6902
Precision	0.2694	0.2685	0.2682	0.2671
F1 Score	0.424	0.423	0.4228	0.4214
RMSE	0.2844	0.2841	0.2837	0.2875

The HMDA database was randomly split into training data (70%) and testing data (30%). The threshold used to convert the continuous response into a binary response is the mean of the outcome variable.

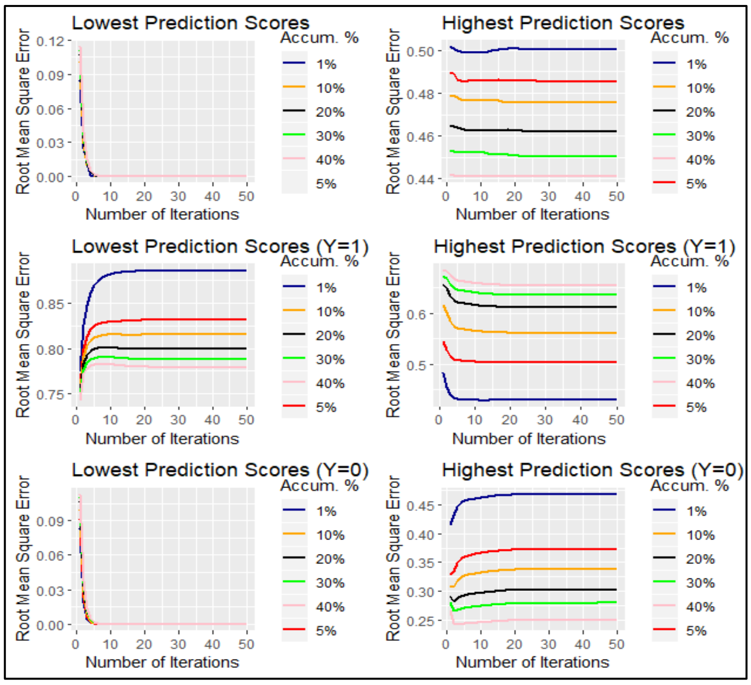

Figure 2. RMSE across iterations of the Synthetic Penalized Logitboost for the HDMA 2012.

Considering the results examined for HMDA, HMDA 2012 and HDMA 2017, the Synthetic Penalized Logitboost increases the true positive rate when predicting a model, in particular in the most extreme observations. And it can reach convergence after some iterations in the boosting procedure.

Table 9. Root-Mean-Square Error of Logistic regression, Logitboost, Gradient Tree Boost and the Synthetic Penalized Logitboost for the training and testing HMDA 2017 data sets.

RMSE for Testing Data Set												
Methods	**Lower Extreme**						**Upper Extreme**					
	0.01	**0.05**	**0.10**	**0.20**	**0.3**	**0.4**	**0.01**	**0.05**	**0.10**	**0.20**	**0.3**	**0.4**
Logistic	0	0	0	0	0	0	0.4266	0.4216	0.4061	0.3957	0.3910	0.3769
Logitboost	0	0	0	0	0	0	0.4282	0.4226	0.4108	0.3991	0.3922	0.3767
Gradient Boost (tree)	0	0	0	0	0	0	0.4514	0.4157	0.4074	0.3969	0.3854	0.3738
Gradient Boost (tree) GS - CV	0	0	0	0	0	0	0.4240	0.4193	0.4098	0.3989	0.3885	0.3768
Synthetic Penalized Logitboost	0	0	0	0	0	0	0.4266	0.4216	0.4061	0.3957	0.3910	0.3769
RMSE for Training Data Set												
Methods	**Lower Extreme**						**Upper Extreme**					
	0.01	**0.05**	**0.10**	**0.20**	**0.3**	**0.4**	**0.01**	**0.05**	**0.10**	**0.20**	**0.3**	**0.4**
Logistic	0	0	0	0	0	0	0.4435	0.4234	0.4115	0.3995	0.3925	0.3768
Logitboost	0	0	0	0	0	0	0.4427	0.4230	0.4115	0.3994	0.3926	0.3768
Gradient Boost (tree)	0	0	0	0	0	0	0.4797	0.4478	0.4234	0.4110	0.3970	0.3762
Gradient Boost (tree) GS - CV	0	0	0	0	0	0	0.4406	0.4227	0.4048	0.3934	0.3856	0.3770
Synthetic Penalized Logitboost	0	0	0	0	0	0	0.4435	0.4234	0.4115	0.3995	0.3925	0.3768

The HMDA database was randomly split into training data (70%) and testing data (30%). Each RMSE for 1%, 5%, 10%, 20%, 30% and 40% of the lowest accumulated prediction scores is shown on the left-hand side of the table under "Lower Extreme", and each RMSE for 1%, 5%, 10%, 20%, 30% and 40% of the highest accumulated prediction scores is shown on the right-hand side of the table under "Upper Extreme". The Gradient Boost (tree) GS – CV is built with 10-fold cross validation and optimized hyperparameters through grid search.

Table 10. Predictive measures of Logistic regression, Logitboost, Gradient Tree Boost and the Synthetic Penalized Logitboost for the testing and training HMDA 2017 data sets.

Testing Data Set				
Predictive Measures	**Logistic Regression**	**Logitboost**	**Gradient Boost (Tree) GS - CV**	**Synthetic Penalized Logitboost**
Recall	0.9983	0.9983	0.9983	0.9981
Specificity	0.6604	0.6604	0.6604	0.6605
Accuracy	0.6839	0.6839	0.6839	0.6840
Precision	0.1805	0.1805	0.1805	0.1805
F1 Score	0.3057	0.3057	0.3057	0.3057
RMSE	0.2385	0.2385	0.2378	0.2387
Training Data Set				
Predictive Measures	**Logistic Regression**	**Logitboost**	**Gradient Boost (Tree) GS - CV**	**Synthetic Penalized Logitboost**
Recall	0.9983	0.9983	0.9983	0.9984
Specificity	0.6637	0.6637	0.6637	0.6637
Accuracy	0.6870	0.6870	0.687	0.687
Precision	0.1817	0.1817	0.1817	0.1817
F1 Score	0.9983	0.3074	0.3074	0.3074
RMSE	0.2382	0.2382	0.2374	0.2383

The HMDA database was randomly split into training data (70%) and testing data (30%). The threshold used to convert the continuous response into a binary response is the mean of the outcome variable.

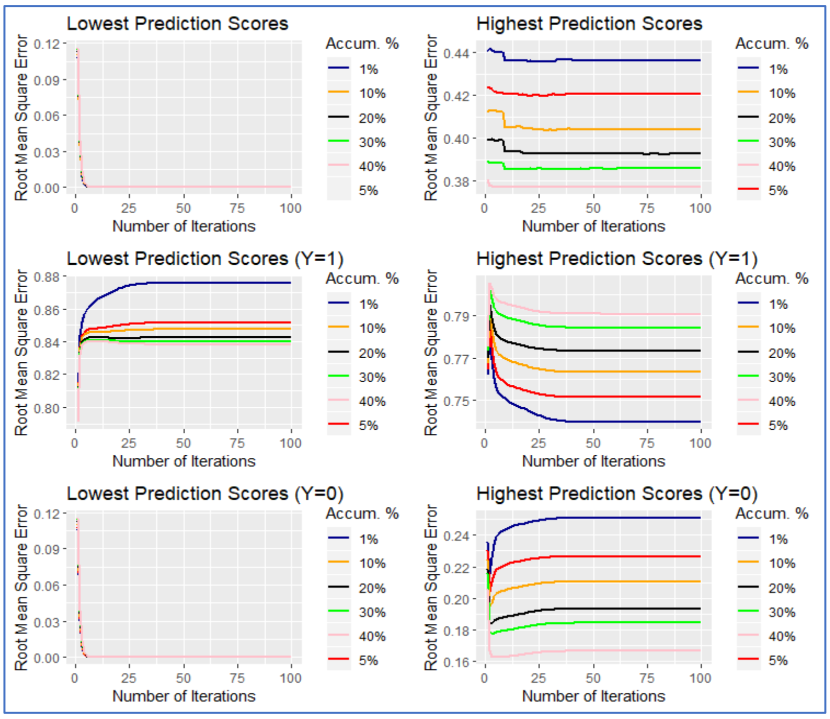

Figure 3. RMSE across iterations of the Synthetic Penalized Logitboost for the HDMA 2017.

References

Barandela, R., Valdovinos, R. M., & Sánchez, J. S. (2003). New applications of ensembles of classifiers. *Pattern Analysis & Applications*, *6*(3), 245-256.

Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). Classification and regression trees. Wadsworth Int. Group, 37(15), 237-251.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321-357.

Dietterich, T. G., Domingos, P., Getoor, L., Muggleton, S., & Tadepalli, P. (2008). Structured machine learning: The next ten years. *Machine Learning*, *73*(1), 3.

Friedman, J. H., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting (with discussion and a rejoinder by the authors). *Annals of Statistics*, *28*(2), 337-407.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 1189-1232.

Gomez-Verdejo, V., Arenas-Garcia, J., Ortega-Moral, M., & Figueiras-Vidal, A. R. (2005). Designing RBF classifiers for weighted boosting. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks* (Vol. 2, pp. 1057-1062). IEEE.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Science & Business Media.

Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analysis*, 6(5):429–449.

King, G., & Zeng, L. (2001). Logistic regression in rare events data. *Political Analysis*, *9*(2), 137-163.

Kotsiantis, S., Kanellopoulos, D., & Pintelas, P. (2006). Handling imbalanced datasets: A review. *GESTS International Transactions on Computer Science and Engineering*, *30*(1), 25-36.

Krawczyk, B. (2016). Learning from imbalanced data: Open challenges and future directions. *Progress in Artificial Intelligence*, *5*(4), 221-232.

Lin, W. C., Tsai, C. F., Hu, Y. H., & Jhang, J. S. (2017). Clustering-based undersampling in class-imbalanced data. *Information Sciences*, *409*, 17-26.

Longadge, R., Dongrre, S.S., & Malik, L. (2013). Class imbalance problem in data mining: Review. *International Journal of Computer Science and Network*, 2(1): 83–87.

McCullagh, P., & Nelder, J. (1989). *Generalized Linear Models*. Chapman and Hall/CRC.

Munnell, A. H., Tootell, G. M., Browne, L. E., & McEneaney, J. (1996). Mortgage lending in Boston: Interpreting HMDA data. *The American Economic Review*, 25-53.

Pesantez-Narvaez, J., Guillen, M., & Alcañiz, M. (2019). Predicting Motor Insurance Claims Using Telematics Data—XGBoost versus Logistic Regression. *Risks*, *7*(2), 70.

Pesantez-Narvaez, J., & Guillen, M. (2020a). Penalized logistic regression to improve predictive capacity of rare events in surveys. *Journal of Intelligent & Fuzzy Systems*, (Preprint), 1-11.

Pesantez-Narvaez J., & Guillen M. (2020b). Weighted Logistic Regression to Improve Predictive Performance in Insurance. *Advances in Intelligent Systems and Computing*, 894, 22-34

Schapire, R. E., & Freund, Y. (2013). Boosting: Foundations and algorithms. *Kybernetes*.

Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., & Napolitano, A. (2009). RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *40*(1), 185-197.

Wang, S., & Yao, X. (2009). Diversity analysis on imbalanced data sets by using ensemble models. In *2009 IEEE Symposium on Computational Intelligence and Data Mining* (pp. 324-331). IEEE.